1.

(a). Average access time = (1 – miss rate) × hit time + miss rate × miss time =0.95 × 1 + 0.05 × 105 = 6.2 cycles.

(b). Because of the randomness of the accesses, the probability and access will be a hit is equal to the size of the cache divided by the size of the array.

Hit rate = 64 Kbytes/256 Mbytes ≈1/4000 = 0.00025.

Therefore, average access time = 0.00025 × 1 + (1 – 0.00025) × 105 = 104.974 cycles.

(c). The access time when the cache is disabled is 100 cycles which is less than the average access time when the cache is enabled and almost all the accesses are misses. If there is no locality at all in the data stream then the cache memory will not only be useless but also it will be a liability.

(d). Assuming the memory access time with no cache is T_off, with cache is T_on and the miss rate is m, the average access time (with cache on)

T_on = (1 – m) (Toff – G) + m (Toff + L)

The cache becomes useless when the miss rate is high enough to make T_off less than or equal to T_on .

At this point we have

T_off < = (1 – m) (Toff – G) + m (Toff + L)

After some algebraic manipulation, the inequality reduces to. For part (a), G=99 and L=5, a miss rate greater than or equal to 99/104 (~ .95) would render the cache useless.

2.

A useful tool for solving this type of problem is to extract all of the available information from the problem description. It is possible that not all of the information
will be necessary to solve the problem, but having it in summary form makes it easier to think about. Here is a summary:

(1) CPU: 1.1 GHz (0.909ns equivalent), CPI of 0.7 (excludes memory accesses)
nstruction mix: 75% non-memory-access instructions, 20% loads, 5%
stores

(2) Caches: Split L1 with no hit penalty, (i.e., the access time is the time it
takes to execute the load/store instruction

(2.1) L1 I-cache: 2% miss rate, 32-byte blocks (requires 2 bus cycles to fill,
miss penalty is 15ns + 2 cycles

(2.2) D-cache: 5% miss rate, write-through (no write-allocate), 95% of all
writes do not stall because of a write buffer, 16-byte blocks (requires 1
bus cycle to fill), miss penalty is 15ns + 1 cycle

(3) L1/L2 bus: 128-bit, 266 MHz bus between the L1 and L2 caches

(4) L2 (unified) cache, 512 KB, write-back (write-allocate), 80% hit rate, 50%
of replaced blocks are dirty (must go to main memory), 64-byte blocks
(requires 4 bus cycles to fill), miss penalty is 60ns + 7.52ns = 67.52ns

(5) Memory, 128 bits (16 bytes) wide, first access takes 60ns, subsequent accesses take 1 cycle on 133 MHz, 128-bit bus

(a)The average memory access time for instruction accesses:

L1 (inst) miss time in L2: 15ns access time plus two L2 cycles (two = 32 bytes in inst. cache line/16 bytes width of L2 bus) = 15 + 2 × 3.75 = 22.5ns. (3.75 is equivalent to one 266 MHz L2 cache cycle)
L2 miss time in memory: 60ns + plus four memory cycles (four = 64 bytes in L2 cache/16 bytes width of memory bus) = 60 + 4 × 7.5 = 90ns (7.5 is equivalent to one 133 MHz memory bus cycle).
Avg. memory access time for inst = avg. access time in L2 cache + avg.access time in memory + avg. access time for L2 write-back.
= 0.02 × 22.5 + 0.02 × (1 – 0.8) × 90 + 0.02 × (1 – 0.8) × 0.5 × 90 = .99ns
(1.09 CPU cycles)

(b)The average memory access time for data reads: Similar to the above formula with one difference: the data cache width is 16 bytes which takes one L2 bus cycles transfer (versus two for the inst. cache), so
L1 (read) miss time in L2: 15 + 3.75 = 18.75ns
L2 miss time in memory: 90ns
Avg. memory access time for read = 0.05 × 18.75 + 0.05 × (1 – 0.8) × 90 + 0.05 × (1 – 0.8) × 0.5 × 90 = 2.29ns (2.52 CPU cycles)

(c)The average memory access time for data writes:
Assume that writes misses are not allocated in L1, hence all writes use the write buffer. Also assume the write buffer is as wide as the L1 data cache.
L1 (write) time to L2: 15 + 3.75 = 18.75ns
L2 miss time in memory: 90ns
Avg. memory access time for data writes = .05 × 18.75 + 0.05 × (1 – 0.8) × 90 + .05 × (1 – 0.8) × 0.5 × 90 = 2.29ns (2.52 CPU cycles)

(d) What is the overall CPI, including memory accesses:
Components: base CPI, Inst fetch CPI, read CPI or write CPI, inst fetch time is added to data read or write time (for load/store instructions).
CPI = 0.7 + 1.09 + 0.2 × 2.52 + 0.05 × 2.52 = 2.42 CPI.

3.

TLB	Page Table
Miss	fault
Hit	hit/X
Miss	fault
Miss	fault
Hit	hit/X
Miss	hit
Hit	hit/X
Miss	hit
Miss	hit
Miss	fault